



# Deep Learning: Restricted Boltzmann Machines & Deep Belief Nets

Based on slides by Geoffrey Hinton, Sue Becker, Yann  
LeCun, Yoshua Bengio, Frank Wood

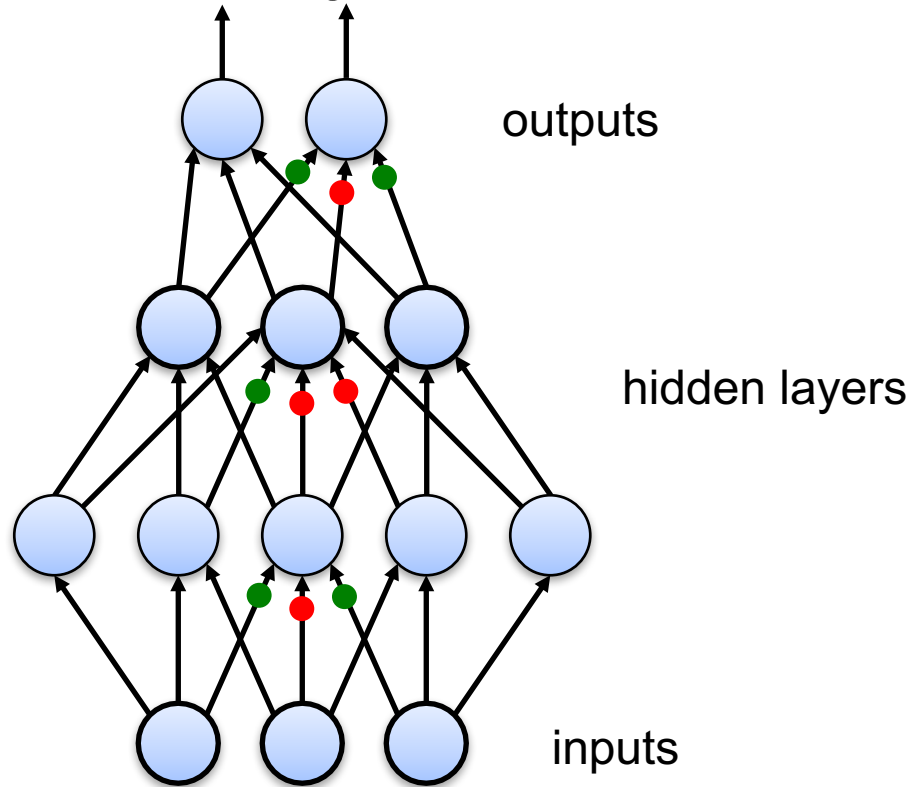
These slides were assembled by Eric Eaton, with grateful acknowledgement of the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution. Please send comments and corrections to Eric.

# Neural Networks

Back-propagate error signal to get derivatives for learning



Compare outputs with correct answer to get error signal



# What is wrong with back-propagation?

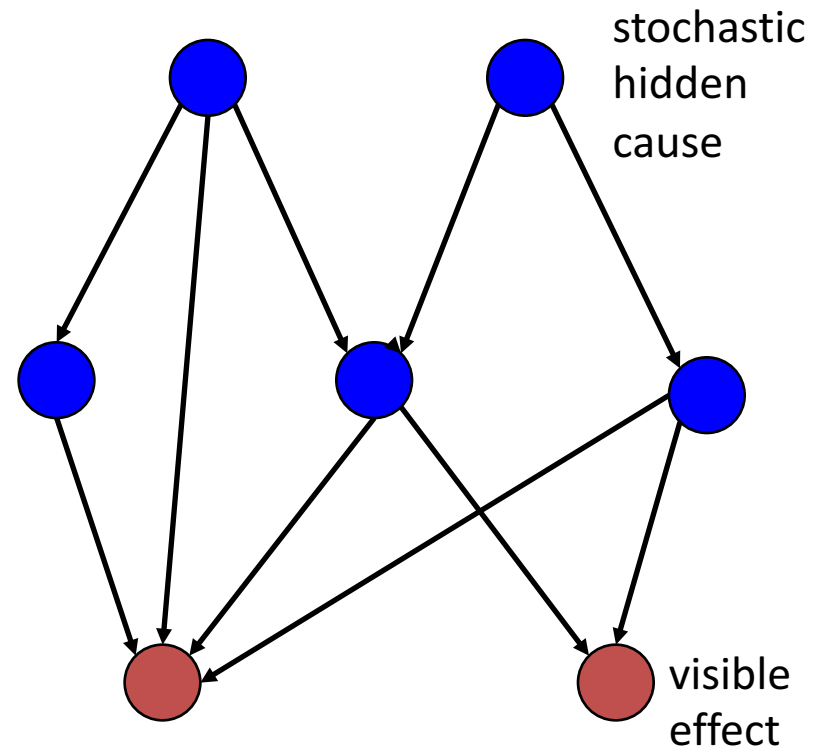
- It requires labeled training data
  - Almost all data is unlabeled
- The learning time does not scale well
  - It is very slow in nets with multiple hidden layers
- It can get stuck in poor local optima
  - These are often quite good, but for deep nets they are far from optimal

# Motivations

- Supervised training of deep models (e.g. many-layered NNets) is difficult (optimization problem)
- Shallow models (SVMs, one-hidden-layer NNets, boosting, etc...) are unlikely candidates for learning high-level abstractions needed for AI
- Unsupervised learning could do “local-learning” (each module tries its best to model what it sees)
- Inference (+ learning) is intractable in directed graphical models with many hidden variables
- Current unsupervised learning methods don’t easily extend to learn multiple levels of representation

# Belief Nets

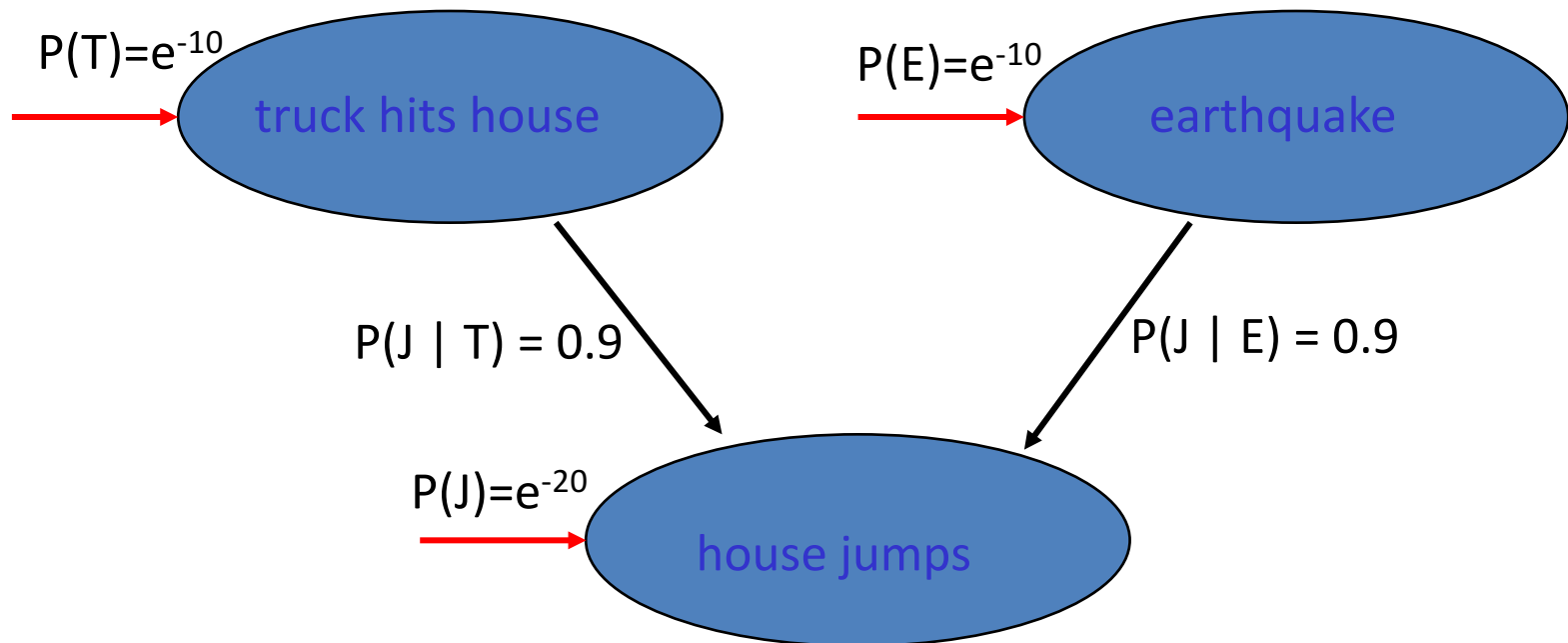
- A belief net is a directed acyclic graph composed of stochastic variables.
- Can observe some of the variables and we would like to solve two problems:
- **The inference problem:**  
Infer the states of the unobserved variables.
- **The learning problem:**  
Adjust the interactions between variables to make the network more likely to generate the observed data.



Use nets composed of layers of stochastic binary variables with weighted connections. Later, we will generalize to other types of variable.

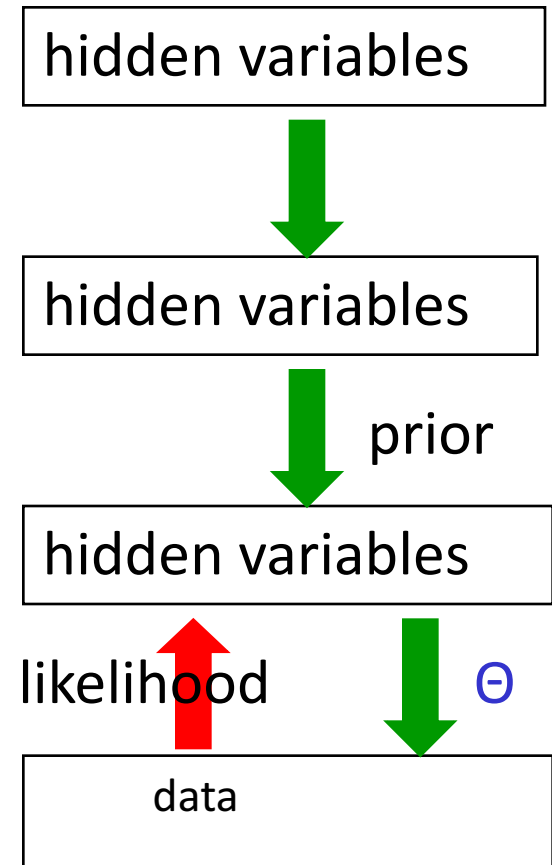
# Explaining away (Judea Pearl)

- Even if two hidden causes are independent, they can become dependent when we observe an effect that they can both influence.
  - If we learn that there was an earthquake it reduces the probability that the house jumped because of a truck.



# Why multilayer learning is hard in a sigmoid belief net

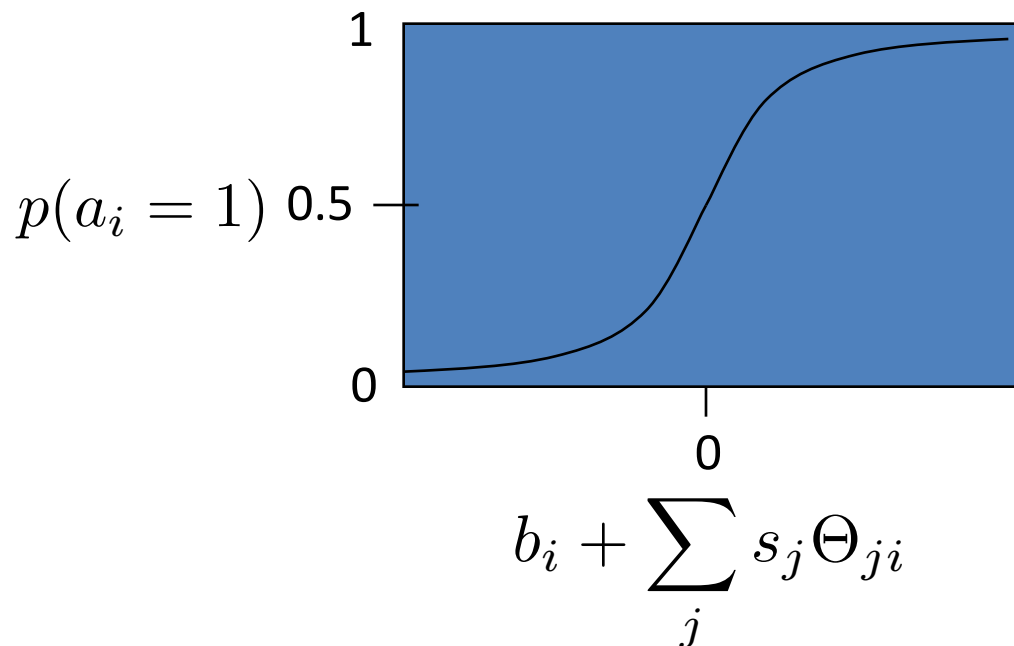
- To learn  $\Theta$ , we need the posterior distribution in the first hidden layer.
- **Problem 1:** The posterior is typically intractable because of “explaining away”.
- **Problem 2:** The posterior depends on the prior created by higher layers as well as the likelihood.
  - So to learn  $\Theta$ , we need to know the weights in higher layers, even if we are only approximating the posterior. **All the weights interact.**
- **Problem 3:** We need to integrate over all possible configurations of the higher variables to get the prior for first hidden layer. **Yuk!**



# Stochastic binary neurons

Have a state of 1 or 0, which is a stochastic function of the neuron's bias  $b$  and the input state  $s$  it receives from other neurons.

$$p(a_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j \Theta_{ji})}$$






# Stochastic units

Replace the binary threshold units by binary stochastic units that make biased random decisions.

- The temperature controls the amount of noise
- Decreasing all the energy gaps between configurations is equivalent to raising the noise level

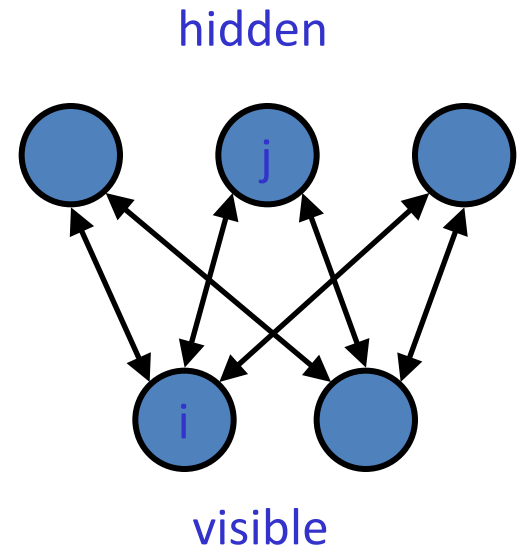
$$P(a_i = 1) = \frac{1}{1 + \exp(-\sum_j s_j \Theta_{ji}/T)} = \frac{1}{1 + \exp(-\Delta E_i/T)}$$

$$\text{Energy gap} = \Delta E_i = E(a_i = 0) - E(a_i = 1)$$

  
temperature

# Restricted Boltzmann Machines

- Restrict the connectivity to make learning easier
  - Only one layer of hidden units
    - Deal with more layers later
  - No connections between hidden units
- In an RBM, the hidden units are conditionally independent given the visible states
  - So can quickly get an unbiased sample from the posterior distribution when given a data-vector
  - This is a big advantage over directed belief nets



# The energy of a joint configuration (ignoring bias terms)

binary state of visible unit  $i$       weight between units  $i$  and  $j$

↓      ↓

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{i,j} v_i h_j \Theta_{ij}$$

↑      ↑

Energy with configuration  $\mathbf{v}$  on the visible units and  $\mathbf{h}$  on the hidden units      binary state of hidden unit  $j$

$$- \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \Theta_{ij}} = v_i h_j$$

# Weights $\rightarrow$ Energies $\rightarrow$ Probabilities

- Each possible joint configuration of the visible and hidden units has an energy
  - The energy is determined by the weights and biases
- The energy of a joint configuration of the visible and hidden units determines its probability:

$$P(\mathbf{v}, \mathbf{h}) \propto e^{-E(\mathbf{v}, \mathbf{h})}$$

- The probability of a configuration over the visible units is found by summing the probabilities of all the joint configurations that contain it.

# Using energies to define probabilities

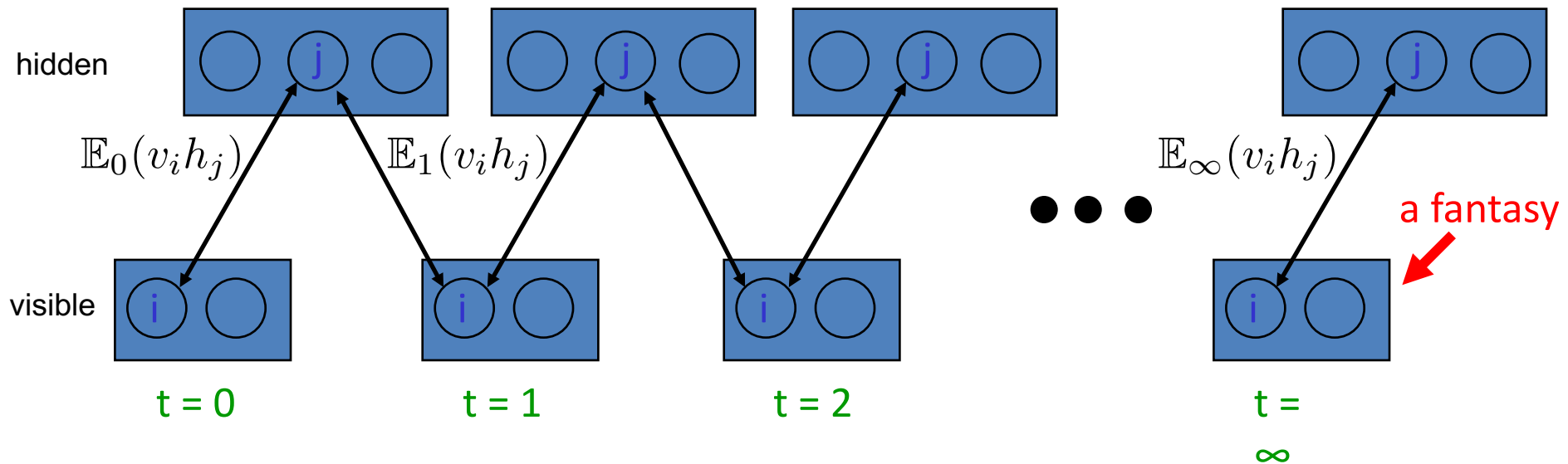
- Probability of a joint configuration over both visible and hidden units

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}$$

- Probability of a particular configuration of the visible units

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}$$

# A picture of the Boltzmann machine learning algorithm for an RBM



Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\frac{\partial \log P(\mathbf{v})}{\partial \Theta_{ij}} = \mathbb{E}_0(v_i h_j) - \mathbb{E}_\infty(v_i h_j)$$

# A very surprising fact

- Everything that one weight needs to know about the other weights and the data in order to do maximum likelihood learning is contained in the difference of two correlations.

$$\frac{\partial \log P(\mathbf{v})}{\partial \Theta_{ij}} = \mathbb{E}_0(v_i h_j) - \mathbb{E}_\infty(v_i h_j)$$



Derivative of log probability of one training vector

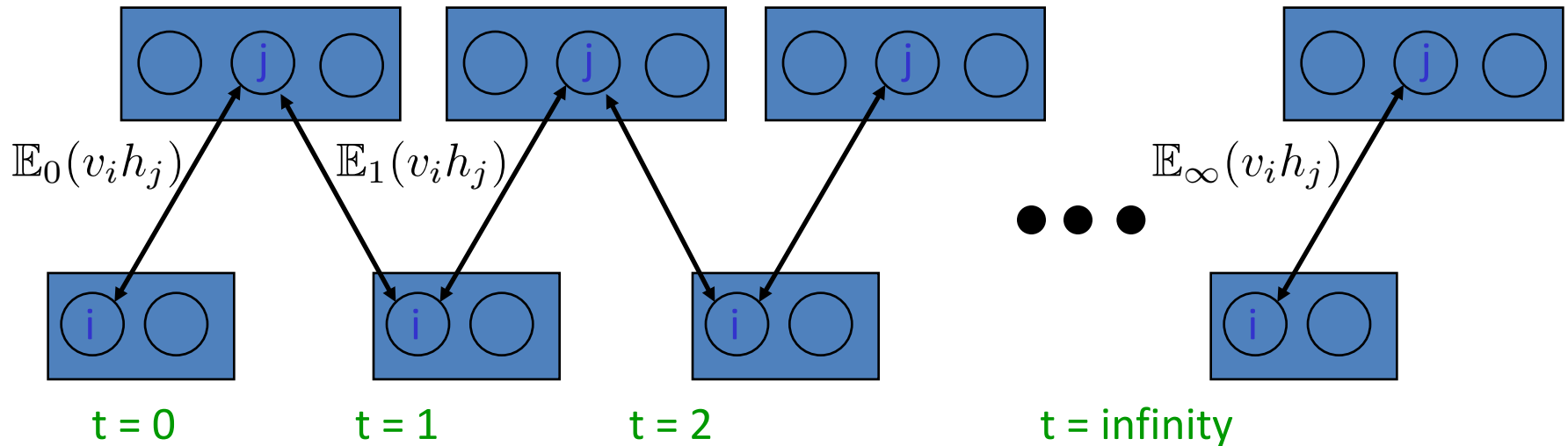


Expected value of product of states at thermal equilibrium when the training vector is clamped on the visible units



Expected value of product of states at thermal equilibrium when nothing is clamped

# A picture of the Boltzmann machine learning algorithm for an RBM

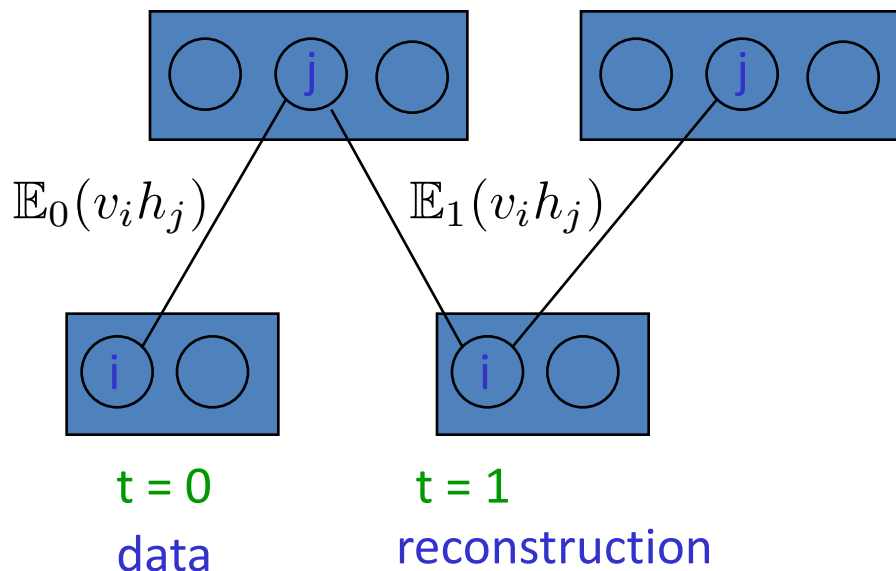


**Problem:** this Markov chain may take a very long time to converge!

**Solution:** Contrastive Divergence



# Contrastive Divergence Learning: A quick way to learn an RBM



Start with a training vector on the visible units.

Update all the hidden units in parallel

Update the all the visible units in parallel to get a “reconstruction”.

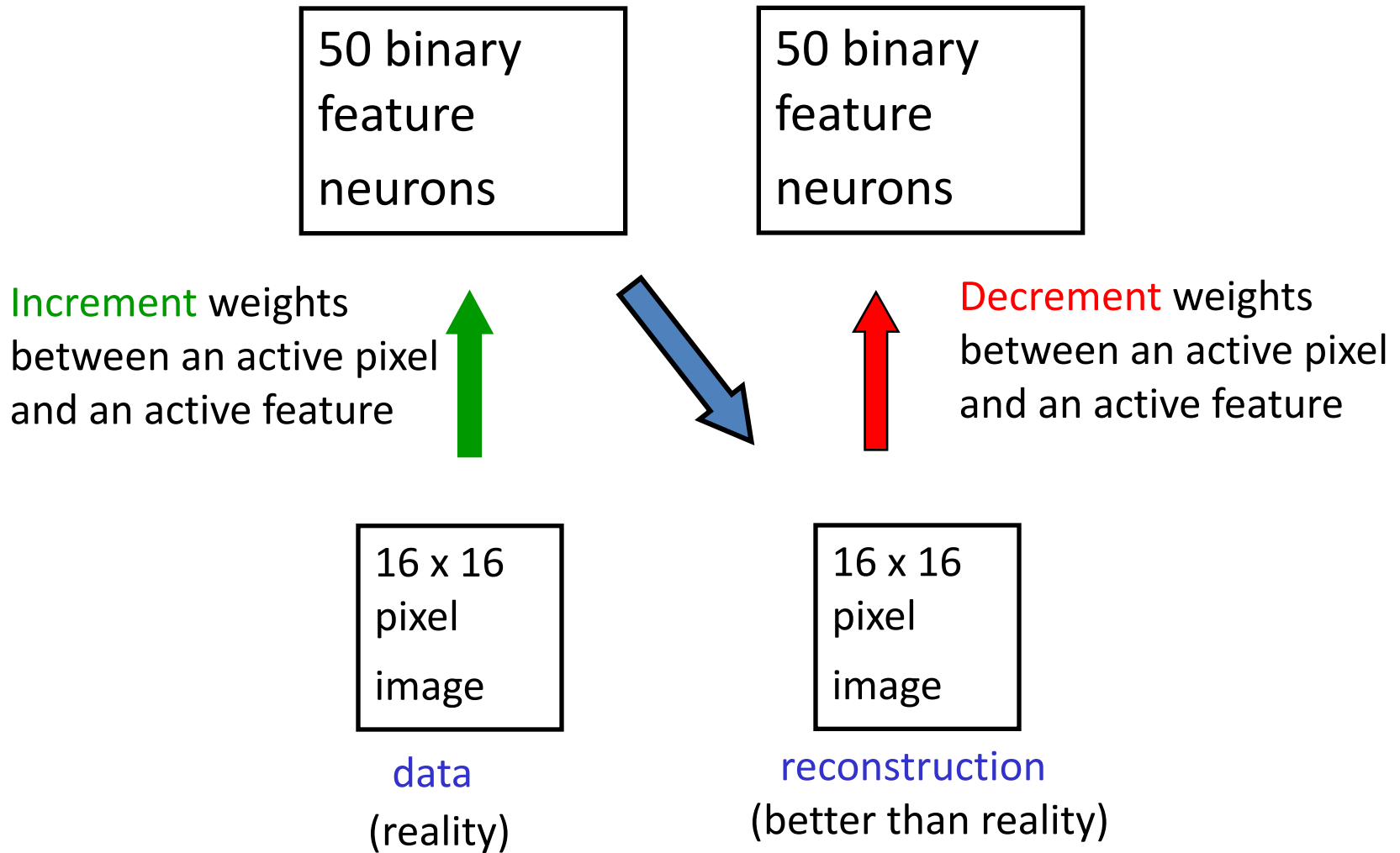
Update the hidden units again.

$$\Delta \Theta_{ij} = \epsilon [\mathbb{E}_0(v_i h_j) - \mathbb{E}_1(v_i h_j)]$$

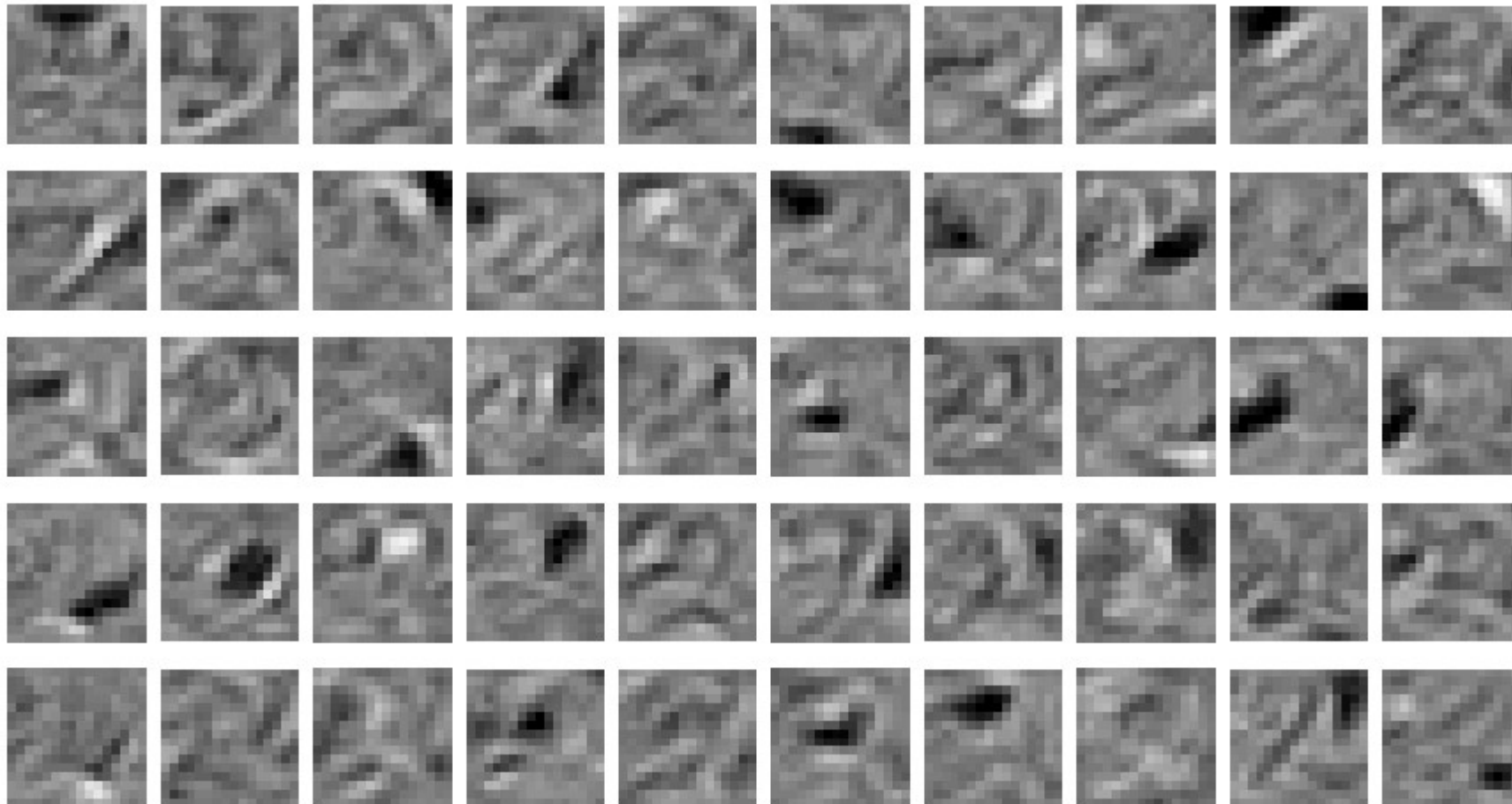
This is not following the gradient of the log likelihood. But it works well.

It is approximately following the gradient of another objective function (Carreira-Perpinan & Hinton, 2005).

# How to learn a set of features that are good for reconstructing images of the digit 2

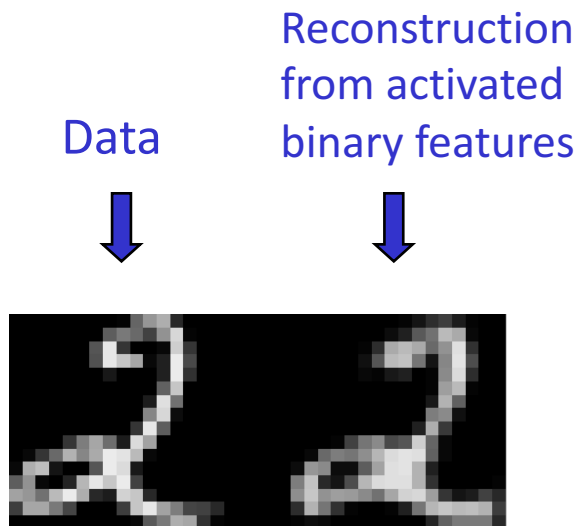


# The Final 50 X 256 Weights

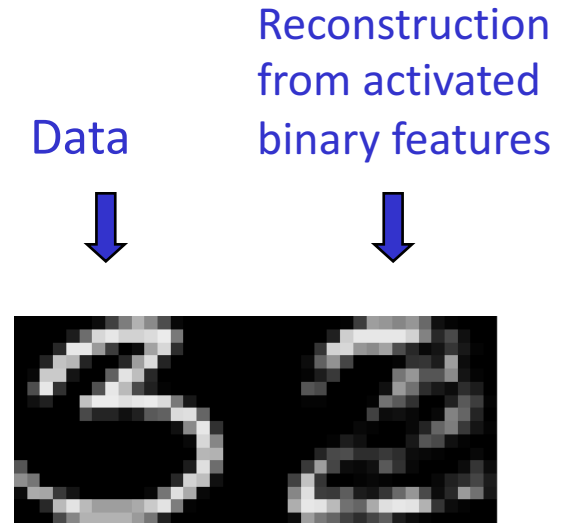


Each neuron grabs a different feature.

# How well can we reconstruct the digit images from the binary feature activations?

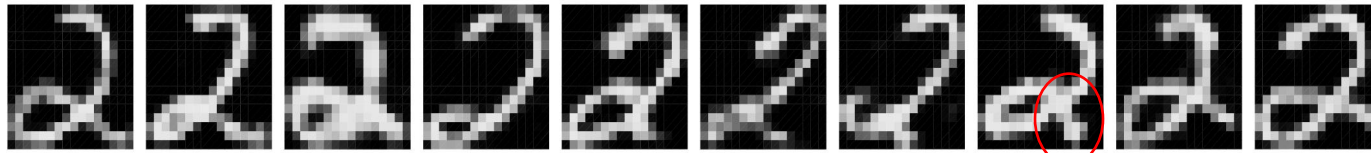


New test images from the digit class that the model was trained on

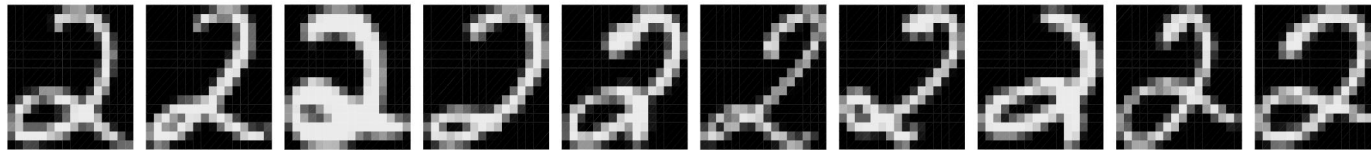


Images from an unfamiliar digit class (the network tries to see every image as a 2)

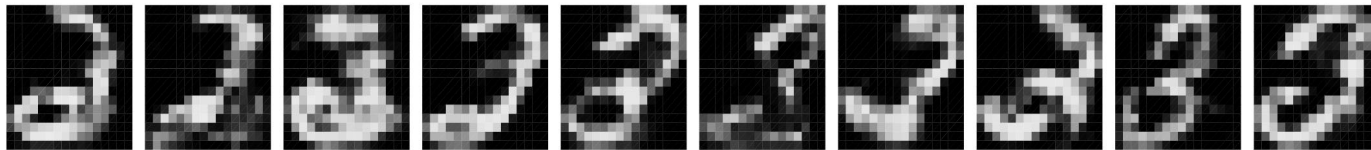
# Using an RBM to learn a model of a digit class



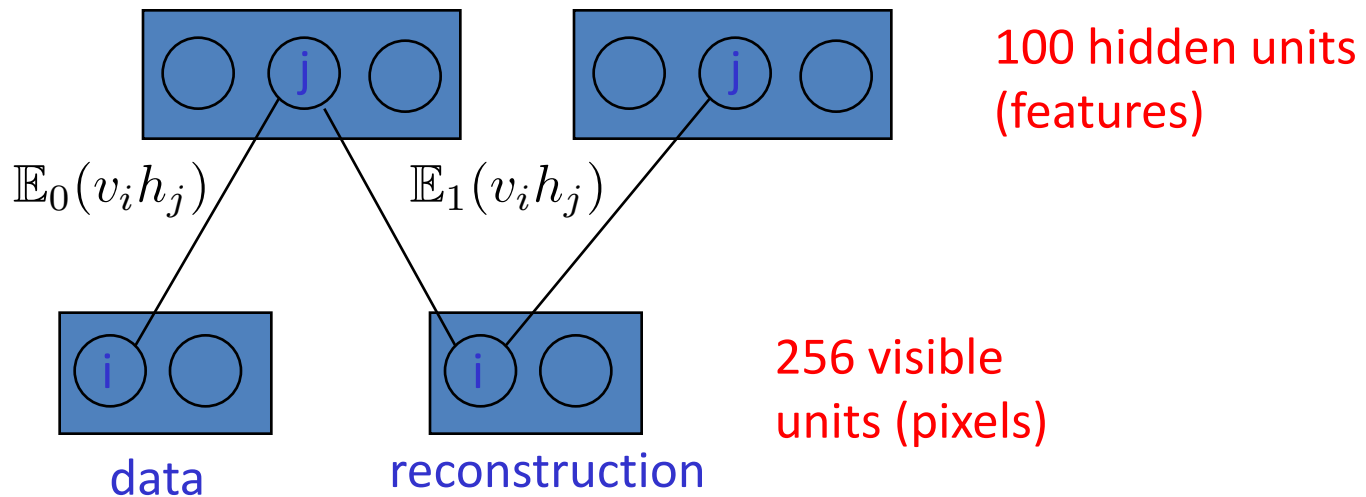
Reconstructions by  
model trained on 2's



Data



Reconstructions by  
model trained on 3's



# Training a Deep Belief Network

(the main reason RBM's are interesting)

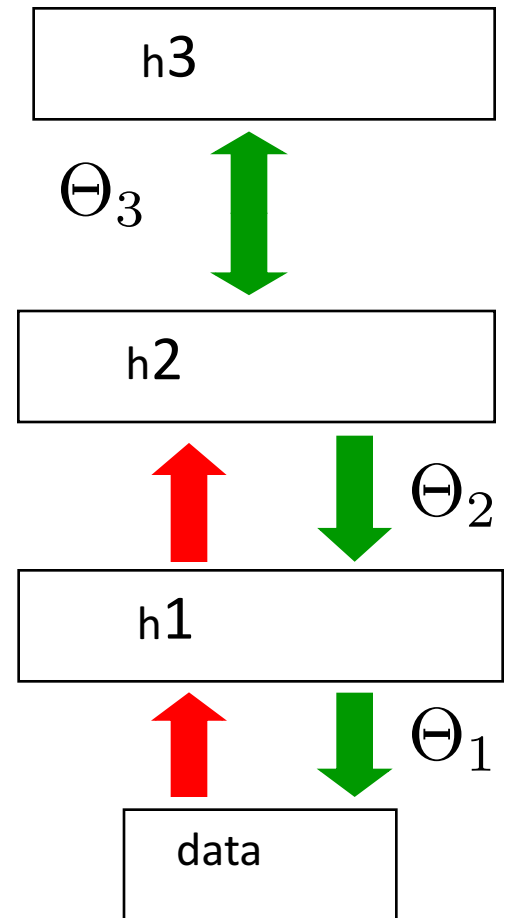
- First train a layer of features that receive input directly from the pixels.
- Then treat the activations of the trained features as if they were pixels and learn features of features in a second hidden layer.
- It can be proved that each time we add another layer of features we improve a variational lower bound on the log probability of the training data.
  - The proof is slightly complicated.
  - But it is based on a neat equivalence between an RBM and a deep directed model

# The Generative Model After Learning 3 Layers

To generate data:

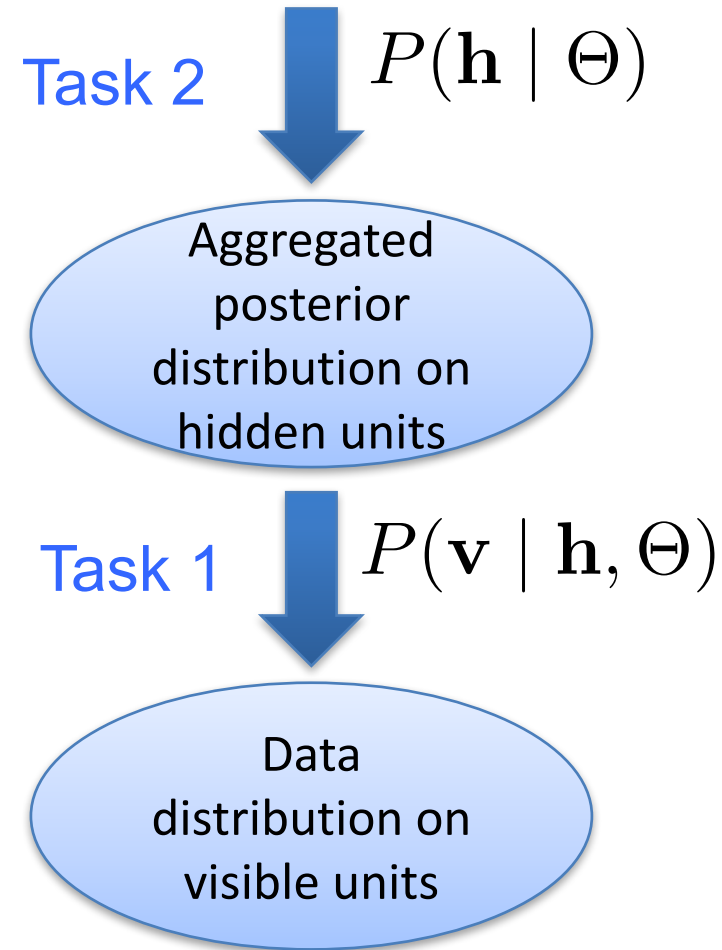
1. Get an equilibrium sample from the top-level RBM by performing alternating Gibbs sampling for a long time.
2. Perform a top-down pass to get states for all the other layers.

So the lower level bottom-up connections are **not** part of the generative model. They are just used for inference.



# Why does greedy learning work?

- Each RBM converts its data distribution into an aggregated posterior distribution over its hidden units.
- This divides the task of modeling its data into two tasks:
  - Task 1: Learn generative weights that can convert the aggregated posterior distribution over the hidden units back into the data distribution.
  - Task 2: Learn to model the aggregated posterior distribution over the hidden units.
  - The RBM does a good job of task 1 and a moderately good job of task 2.
- Task 2 is easier (for the next RBM) than modeling the original data because the aggregated posterior distribution is closer to a distribution that an RBM can model perfectly.





# Why does greedy learning work?

- The weights  $\Theta$  in the bottom level RBM define  $P(\mathbf{v} | \mathbf{h})$  and they also, indirectly, define  $P(\mathbf{h})$ .
- So we can express the RBM model as

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v} | \mathbf{h}, \Theta) P(\mathbf{h} | \Theta)$$

- If we leave  $P(\mathbf{v} | \mathbf{h}, \Theta)$  alone and improve  $P(\mathbf{h} | \Theta)$ , we will improve  $P(\mathbf{v})$ .
- To improve  $P(\mathbf{h})$ , we need it to be a better model of the aggregated posterior distribution over hidden vectors produced by applying  $\Theta$  to the data.
  - Accomplished by the next higher layer

# Why greedy learning works

- Each time we learn a new layer, the inference at the layer below becomes incorrect, but the variational bound on the log prob of the data improves (only true in theory)
- Since the bound starts as an equality, learning a new layer never decreases the log prob of the data, provided we start the learning from the tied weights that implement the complementary prior
- Now that we have a guarantee we can loosen the restrictions and still feel confident
  - Allow layers to vary in size
  - Do not start the learning at each layer from the weights in the layer below

# A neural network model of digit recognition

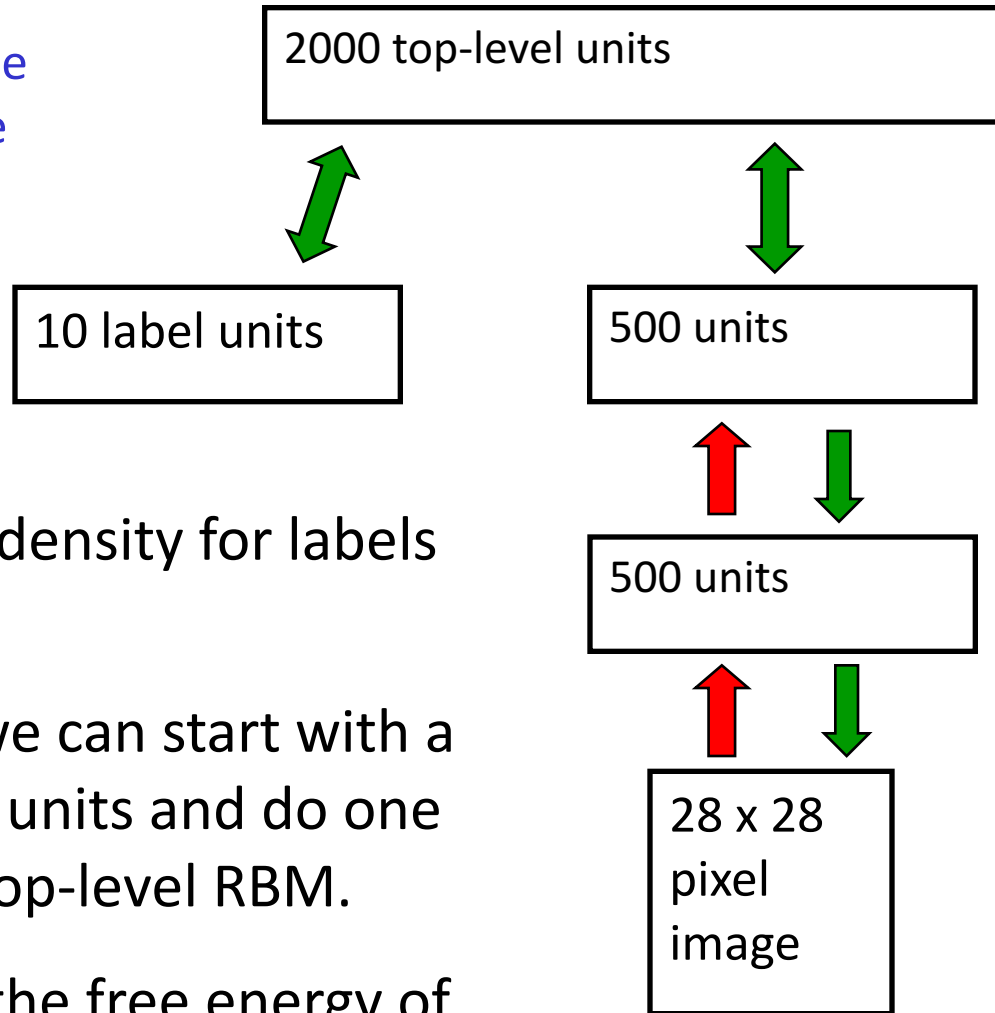
The top two layers form a restricted Boltzmann machine whose free energy landscape models the low dimensional manifolds of the digits.

The valleys have names:

The model learns a joint density for labels and images.

To perform recognition we can start with a neutral state of the label units and do one or two iterations of the top-level RBM.

Or we can just compute the free energy of the RBM with each of the 10 labels



# Movie of the network generating digits

(available at [www.cs.toronto/~hinton](http://www.cs.toronto/~hinton))

# Fine-tuning with a contrastive version of the “wake-sleep” algorithm

After learning many layers of features, we can fine-tune the features to improve generation.

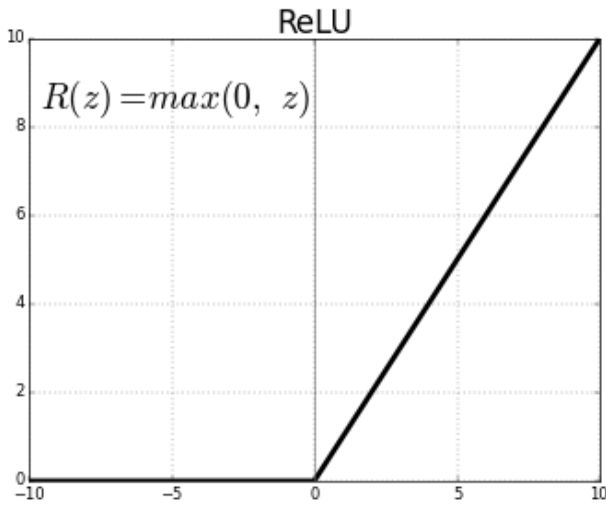
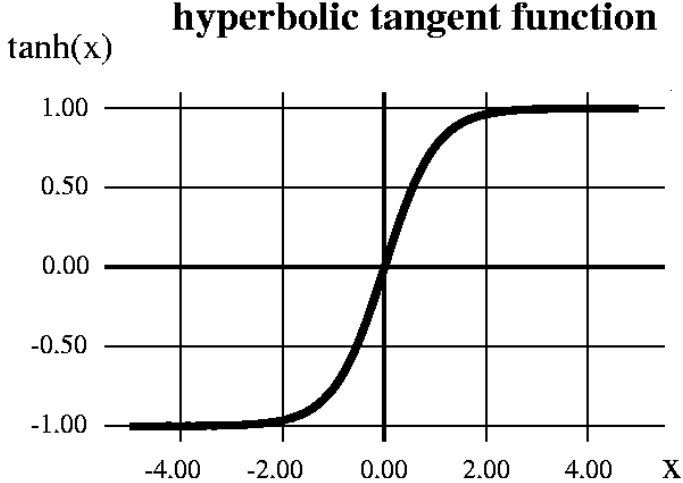
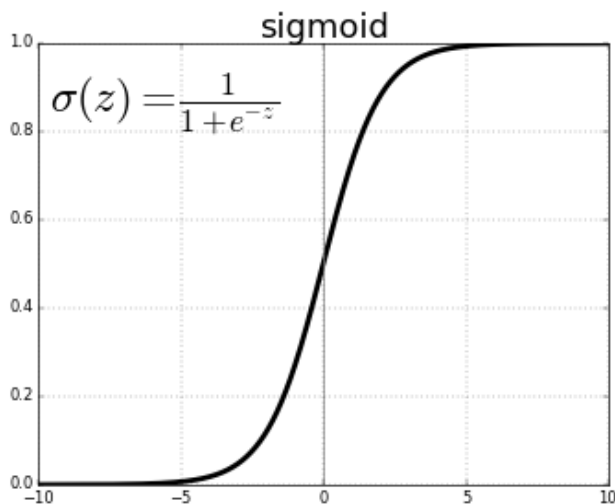
1. Do a stochastic bottom-up pass
  - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. Do a few iterations of sampling in the top level RBM
  - Adjust the weights in the top-level RBM.
3. Do a stochastic top-down pass
  - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.

**Not required! But helps the recognition rate.**

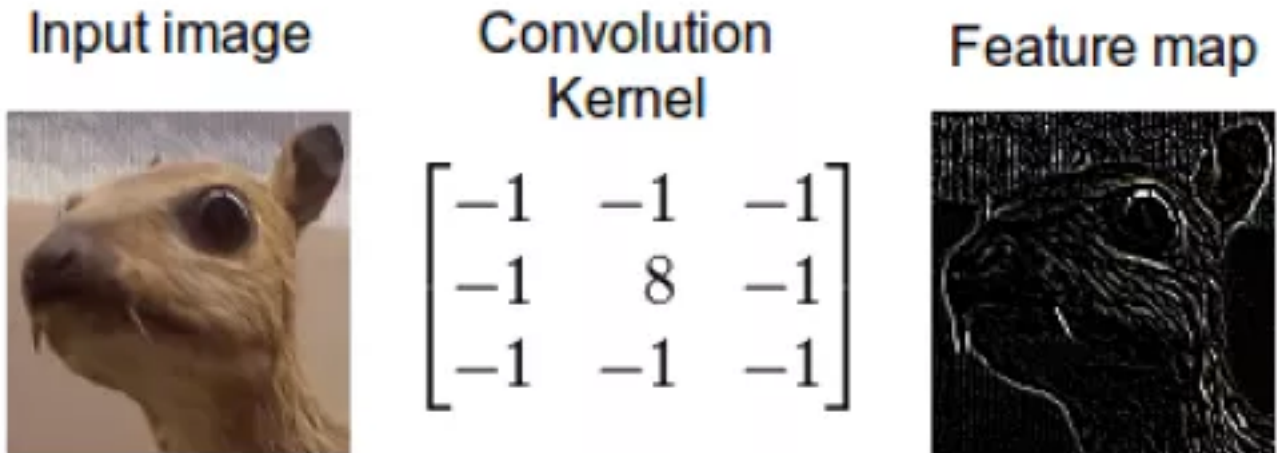
# Limits of the Generative Model

1. Designed for images where non-binary values can be treated as probabilities.
2. Top-down feedback only in the highest (associative) layer.
3. No systematic way to deal with invariance.
4. Assumes segmentation already performed and does not learn to attend to the most informative parts of objects.

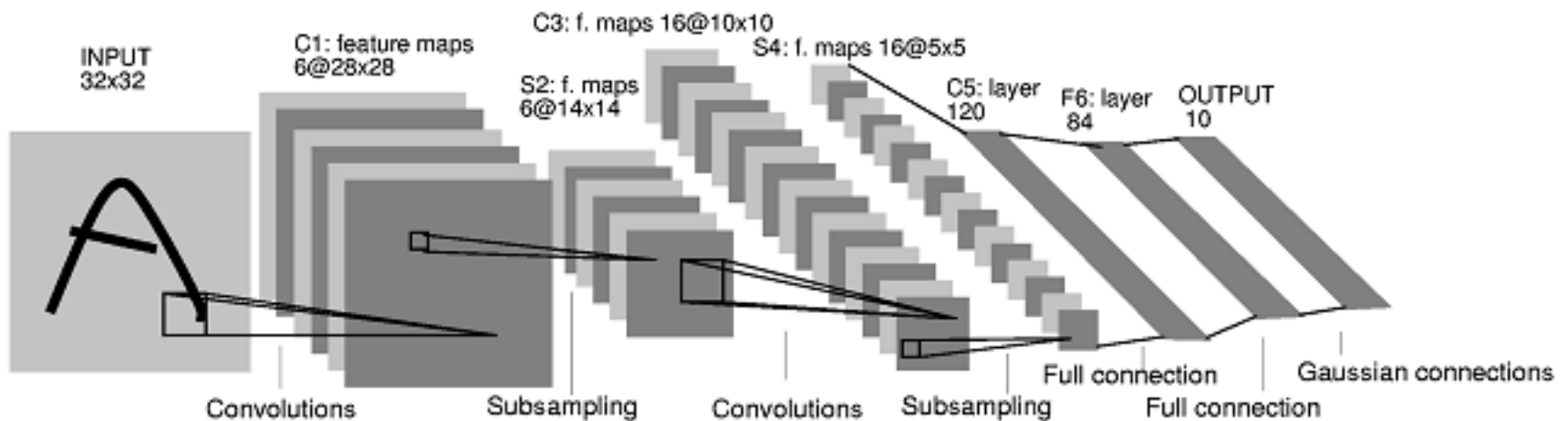
# Deep Net Activation Functions



# Other Deep Architectures: Convolutional Neural Network

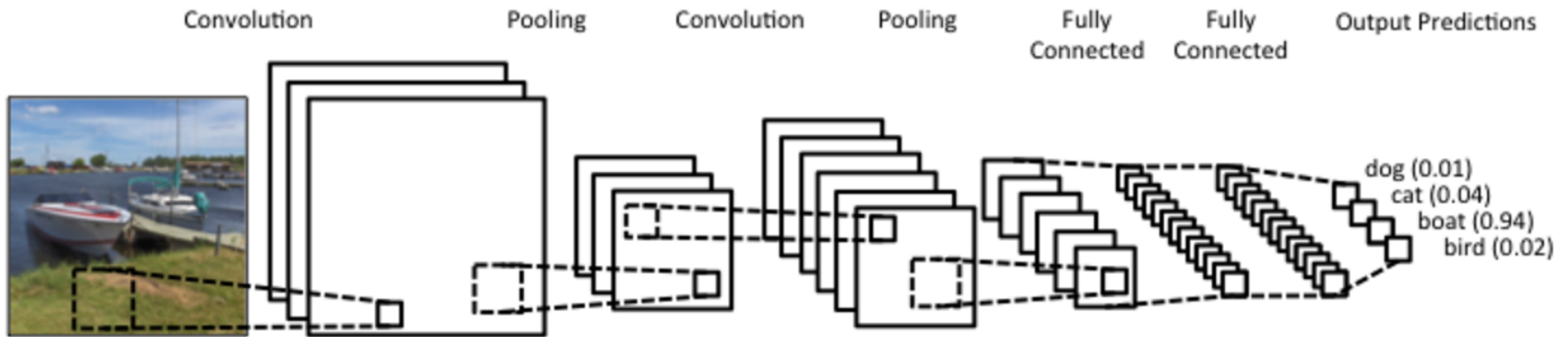


[Image credit: <http://timdettmers.com/2015/03/26/convolution-deep-learning/>]

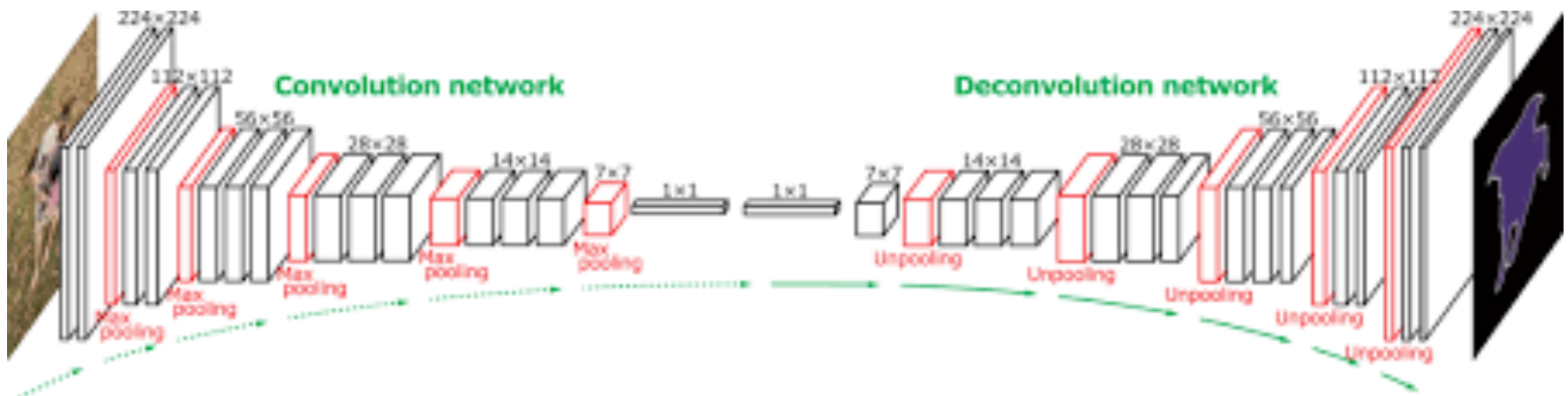




# Other Deep Architectures: Convolutional Neural Network

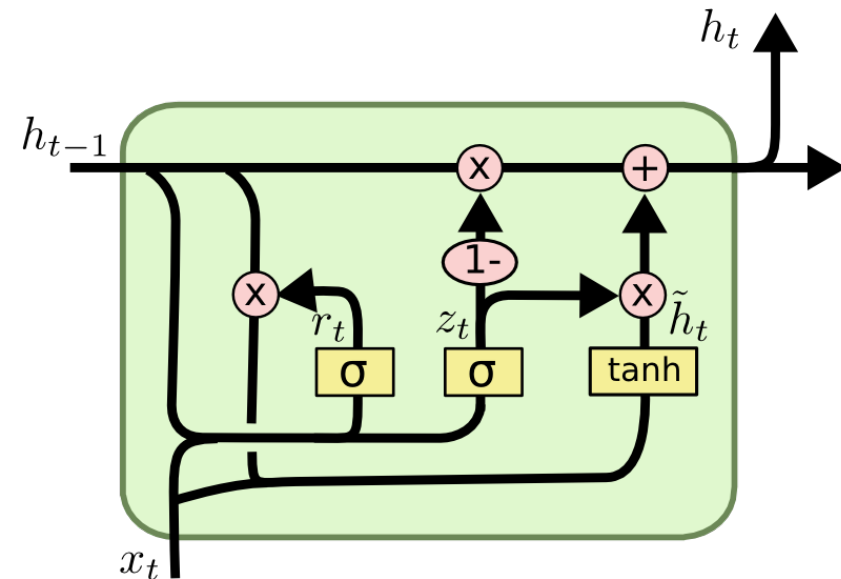
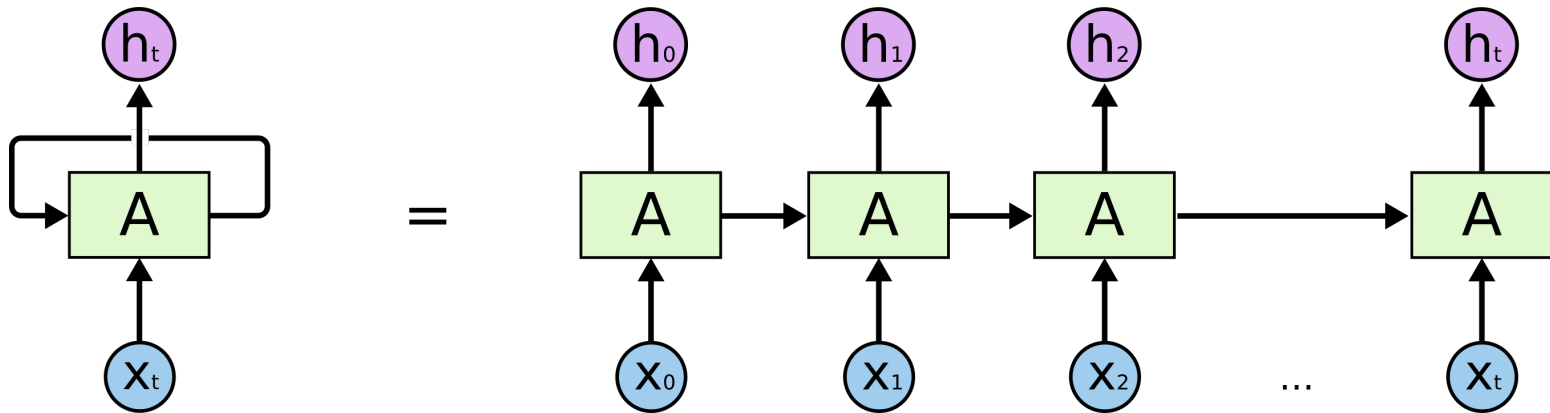


[Image credit: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>]



[Image credit: [http://rnd.azoft.com/wp-content/uploads\\_rnd/2016/11/overall-1024x256.png](http://rnd.azoft.com/wp-content/uploads_rnd/2016/11/overall-1024x256.png)]

# Other Deep Architectures: Long Short-Term Memory (LSTM)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

# Deep Learning in the Headlines

BUSINESS NEWS

MIT  
Technology  
Review

## Is Google Cornering the Market on Deep Learning?

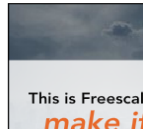
A cutting-edge corner of science is being wooed by Silicon Valley, to the dismay of some academics.

By Antonio Regalado on January 29, 2014



How much are a dozen deep-learning researchers worth? Apparently, more than \$400 million.

This week, Google [reportedly paid that much](#) to acquire [DeepMind Technologies](#), a startup based in



## BloombergBusinessweek Technology

Acquisitions

## The Race to Buy the Human Brains Behind Deep Learning Machines

By Ashlee Vance | January 27, 2014

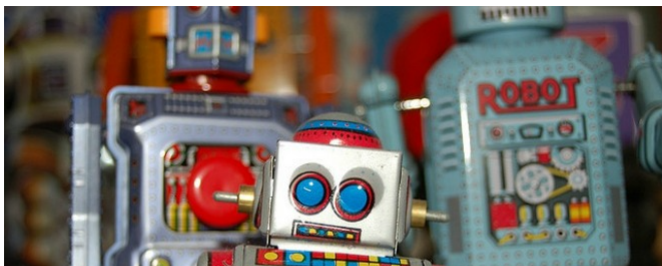
intelligence projects. “DeepMind is bona fide in terms of its research capabilities and depth,” says Peter Lee, who heads Microsoft Research.

According to Lee, Microsoft, Facebook (FB), and Google find themselves in a battle for deep learning talent. Microsoft has gone from four full-time deep learning experts to 70 in the past three years. “We would have more if the talent was there to

**WIRED** GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN  
INNOVATION INSIGHTS | community content | featured

## Deep Learning's Role in the Age of Robots

BY JULIAN GREEN, JETPAC 05.02.14 2:56 PM



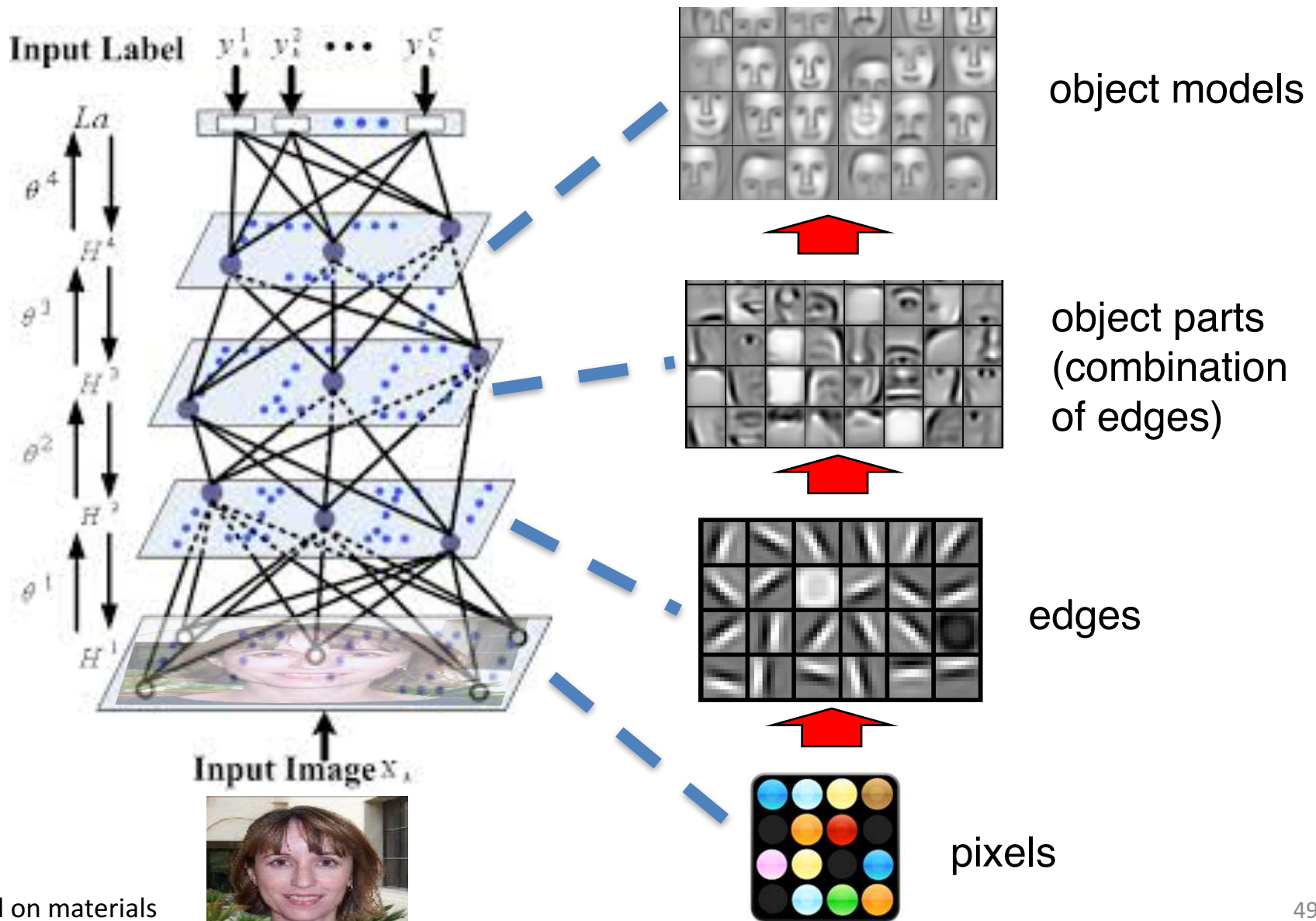
**DEEP LEARNING**

- » Computers learning and growing on their own
- » Able to understand complex, massive amounts of data

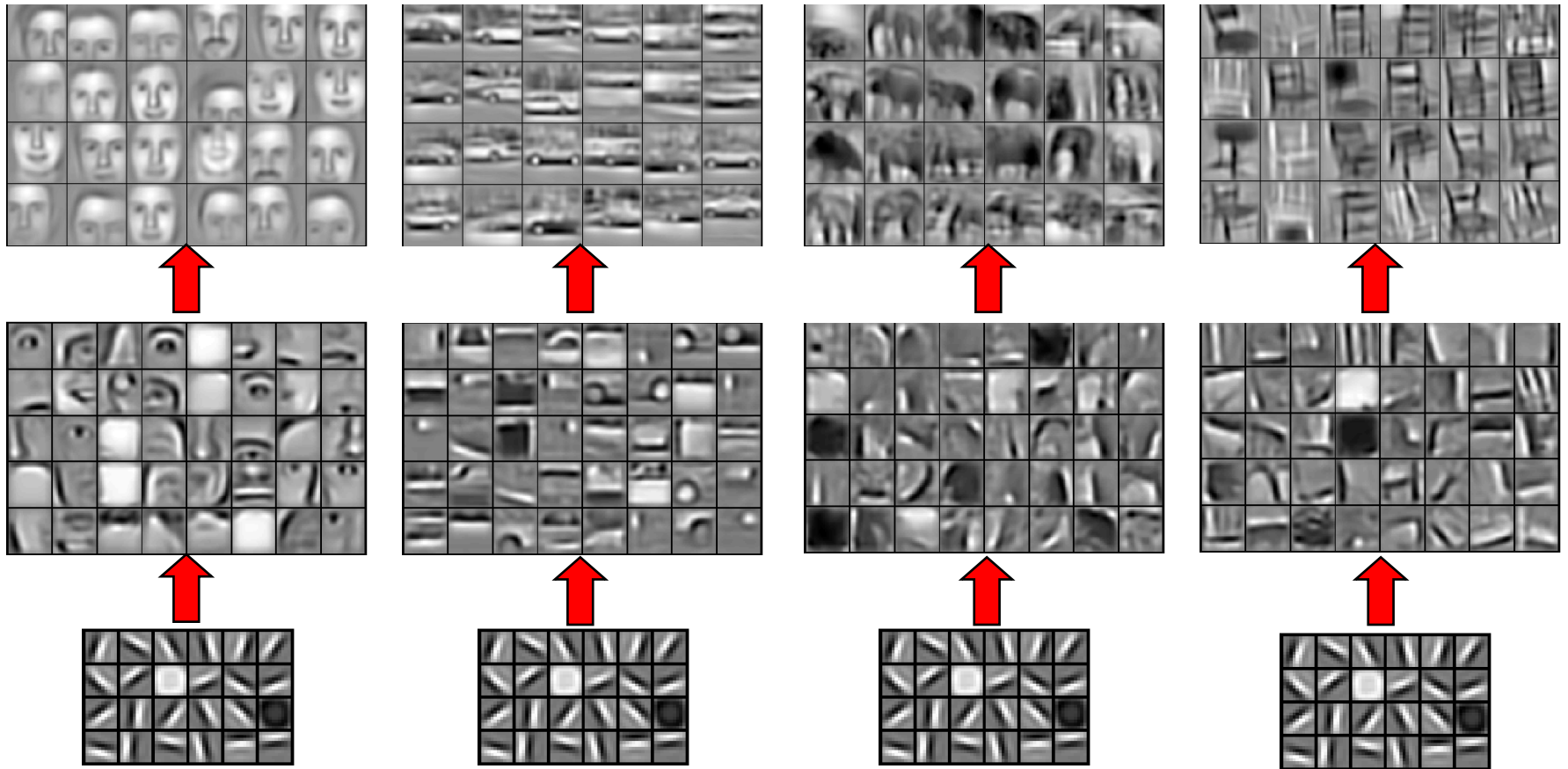
DATA ECONOMY  
DEEP LEARNING

BROUGHT TO YOU BY:

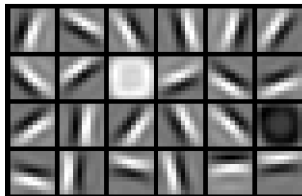
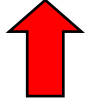
# Deep Belief Net on Face Images



# Learning of Object Parts



# Training on Multiple Objects



Trained on 4 classes (cars, faces, motorbikes, airplanes).

Second layer: Shared-features and object-specific features.

Third layer: More specific features.



# Inference from Deep Learned Models

Generating posterior samples from faces by “filling in” experiments (cf. Lee and Mumford, 2003). Combine bottom-up and top-down inference.

Input images



Samples from feedforward Inference (control)



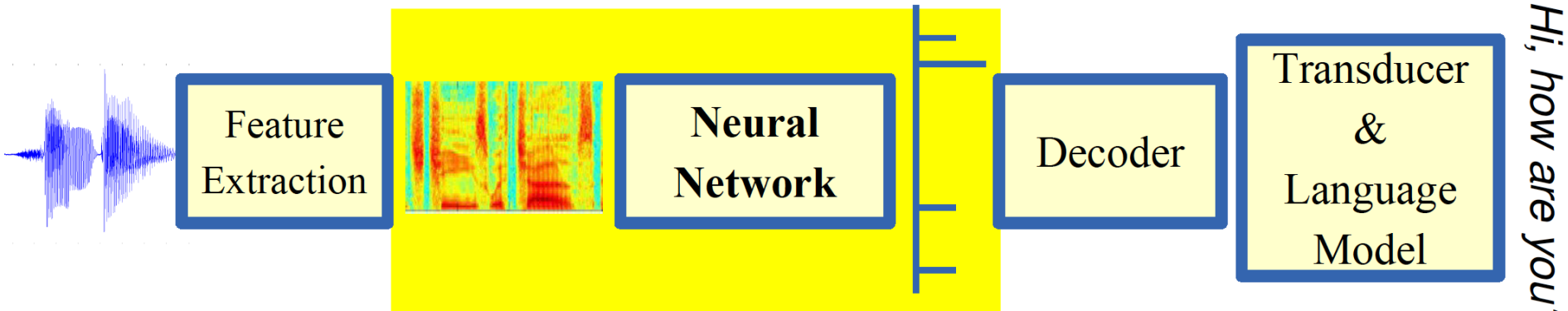
Samples from Full posterior inference



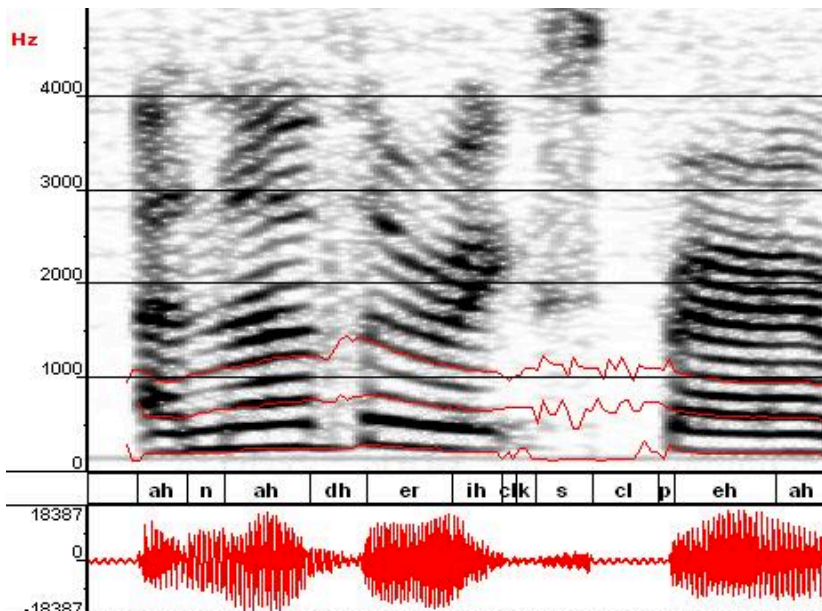


# Machine Learning in Automatic Speech Recognition

## A Typical Speech Recognition System



ML used to predict of phone states from the sound spectrogram



Deep learning has state-of-the-art results

# Hidden Layers	1	2	4	8	10	12
Word Error Rate %	16.0	12.8	11.4	10.9	11.0	11.1

Baseline GMM performance = 15.4%

[Zeiler et al. "On rectified linear units for speech recognition" ICASSP 2013]

# Impact of Deep Learning in Speech Technology

